Software Implementation of the IEEE 754R Decimal Floating-Point Arithmetic

Marius Cornea and Cristina Anderson

July 10-12, 2006

Decimal Applications in the Industry

- Applications that involve financial computations: banking, telephone billing, tax calculation, currency conversion, insurance, accounting in general
- Current feedback indicates that decimal computations take a small fraction of the total execution time
- No indication that scientific computation will migrate to decimal arithmetic in the near future
- IEEE 754R addresses the need for good quality decimal arithmetic, and defines three basic formats: decimal32, decimal64, decimal128
- IEEE 754 example of 'decimal' computation and output:

```
float a = 7/10000.0, b = 10000.0 * a;
printf ("a = %x = %10.10f\n", *(unsigned int *)&a, a);
printf ("b = %x = %10.10f\n", *(unsigned int *)&b, b);

a = 3a378034 = 0.0007000000
b = 40dfffff = 6.9999997504
```

- Error fixed by: decimal32 a = 7/10000.0, b = 10000.0 * a;

Encoding Options: HW or SW?

- Example for decimal64 values v = $(-1)^s \cdot$ coefficient $\cdot 10^{exponent}$ (up to 16 decimal digits; exponent range = [-383, 384]; exp. bias = 383)
- DPD = Densely Packed Decimal is the basis for the IEEE 754R decimal encoding; up to three decimal digits are encoded in 10-bit fields named declets (non-linear mapping); the encoding is "s G E T":
  - s = 1-bit sign
  - G = $G_0G_1G_2G_3G_4$ is 5-bit combination field: encodes the leading decimal digit and the top two exponent bits; when G is 110xx or 1110x then $d_0 = 8+G_4$ and the two top exponent bits are $2G_2+G_3$; when G is 0xxxx or 10xxx then $d_0 = 4G_2+2G_3+G_4$ and the two top exponent bits are $2G_0+G_1$
  - E = 8-bit exponent field, representing the lower 8 bits of the biased exponent
  - T = 50 lower coefficient bits (significand) consisting of 5 declets
- BID = Binary Integer Decimal (the coefficient c is a binary integer) represents the IEEE 754R binary encoding: "s E c_{52-0}" if the coefficient $d_0d_1\ldots d_{15}$ represented as a binary integer fits in 53 bits or "s 11 E c_{50-0}" otherwise (then $c_{53-51} = 100$); E takes 10 bits
- BID does not require conversion to/from binary format on binary HW

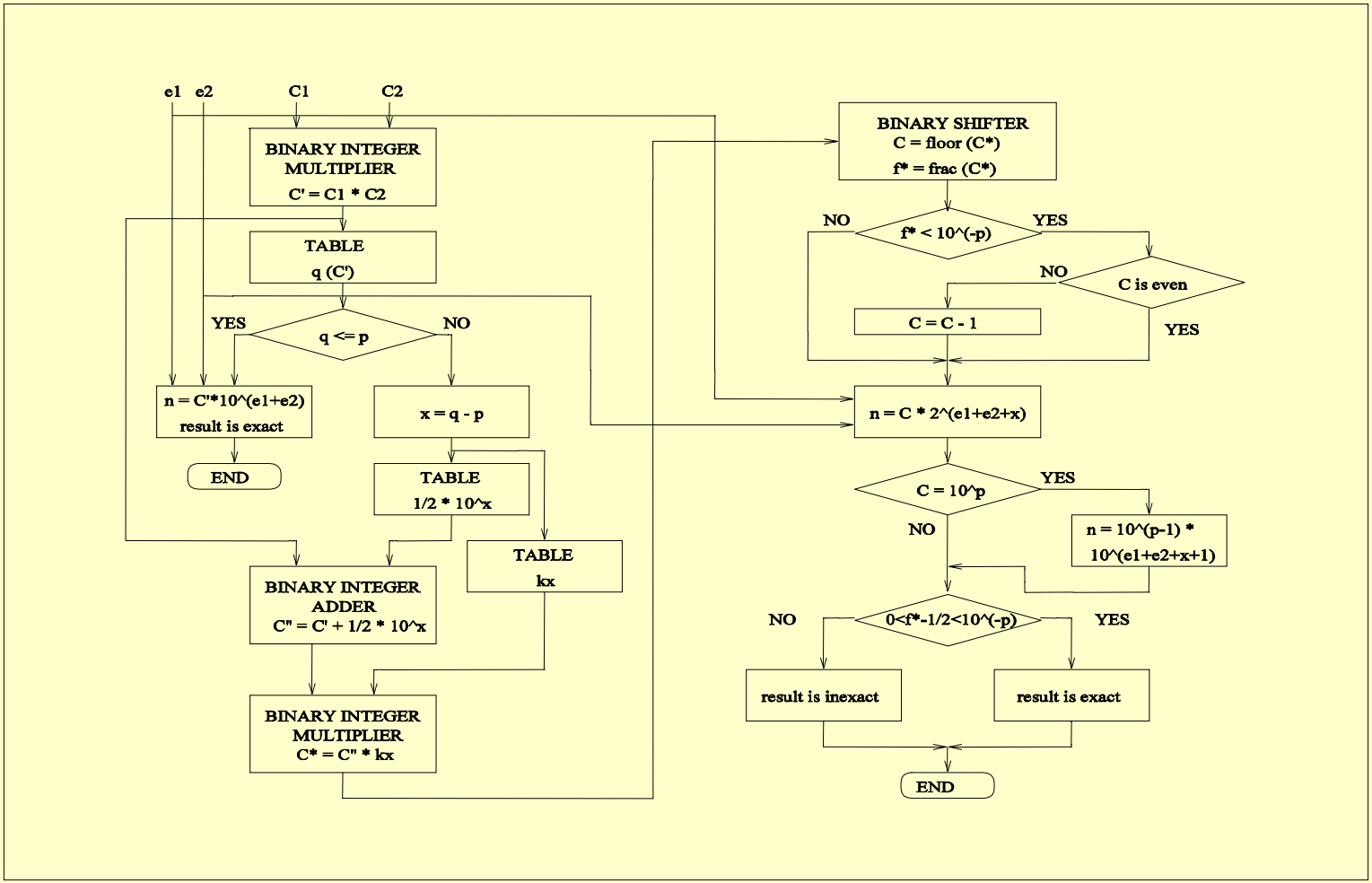
Basic Property for Decimal FP Arithmetic on Binary Hardware

- Property: Let $C \in N$ be a number in base b = 2 and
  $C = d_0 \cdot 10^{q-1} + d_1 \cdot 10^{q-2} + d_2 \cdot 10^{q-3} + \ldots + d_{q-2} \cdot 10^1 + d_{q-1}$
  its representation in base B=10, where $d_0, d_1, \ldots d_{q-1} \in \{0, 1, \ldots, 9\}$ and $d_0 \neq 0$.
  Let $x \in \{1, 2, 3, \ldots, q-1\}$ and $\rho = \log_2 10$.
  If $y \in N$, $y \geq \lceil \{\rho \cdot x\} + \rho \cdot q \rceil$ and k_x is the value of 10^{-x} rounded up to y bits (the subscript RP,y indicates rounding up to y bits in the significand), i.e.:
  $$k_x = (10^{-x})_{RP,y} = 10^{-x} \cdot (1 + \varepsilon) \qquad 0 < \varepsilon < 2^{-y+1}$$
  then $\lfloor C \cdot k_x \rfloor = d_0 \cdot 10^{q-x-1} + d_1 \cdot 10^{q-x-2} + \ldots + d_{q-x-2} \cdot 10^1 + d_{q-x-1}$
- Example: If C = 123456789 is available in binary and its six most significant decimal digits are required, then q = 9 and thus the constant $k_3 \approx 10^{-3}$ must be calculated to y = 31 bits in order to have $\lfloor C \cdot k_3 \rfloor = 123456$ with certainty

Software Implementation of the IEEE 754R Decimal FP Arithmetic

- The values k_x for all x of interest are pre-calculated and are stored as pairs (K_x, e_x) with K_x and e_x positive integers, and $k_x = K_x \cdot 2^{-ex}$.
- Several decimal floating-point operations, in particular addition, subtraction, multiplication, fused multiply-add, and most conversions can thus be implemented efficiently using operations in the integer domain
- An important property is that when rounding the exact result to p digits, the information necessary to determine whether the result is exact (in the IEEE 754 sense) or perhaps a midpoint is available in the product $C \cdot k_x$ itself
- For division and square root, the algorithms are based on scaling the operands so as to bring the results into desired integer ranges, in conjunction with a few floating-point operations and one or two refinement iterations
- The algorithms and operations mentioned here represent the core of a generic implementation in C of the IEEE 754R decimal floating-point arithmetic

Example: Decimal floating-point multiplication with rounding to nearest using hardware for binary operations. From $n1 = C1 \cdot 10^{e1}$ and $n2 = C2 \cdot 10^{e2}$ the product $n = (n1 \cdot n2)_{RN,p} = C \cdot 10^{e}$ is calculated.



Performance Results: BID Library versus decNumber on EM64t for 64-bit Operations

Operation	New BID Library (binary encoding) [clock cycles]				decNumber Library, GCC 4.2 (decimal encoding) [clock cycles]				decNumber (DPD) time / new library (BID) time			
	MIN	AVG	MED	MAX	MIN	AVG	MED	MAX	MIN	AVG	MED	MAX
add	18	119	141	240	423	1200	1310	1674	5.58	11.8	10.5	35.6
mul	33	141	148	215	518	1100	1190	1910	5.78	9.36	7.88	21.9
div	169	314	333	488	627	2320	2290	3390	3.70	7.29	7.13	10.7
sqrt	116	241	247	288	11500	15200	16100	18200	53.5	69.2	59.2	102
to string	53	166	155	254	198	519	492	1060	1.97	3.11	3.18	4.53
from string	9	183	203	288	209	574	550	1190	2.39	2.99	2.98	4.12
cmp	4	47	73	151	348	1050	1060	1530	5.47	27.9	17.1	280
quantize	65	78	72	98	364	611	544	1060	4.97	7.96	6.74	12.18

Performance Results: BID Library versus decNumber on EM64t for 128-bit Operations

Operation	New BID Library (binary encoding) [clock cycles]				decNumber Library, GCC 4.2 (decimal encoding) [clock cycles]				decNumber (DPD) time /new library (BID) time			
	MIN	AVG	MED	MAX	MIN	AVG	MED	MAX	MIN	AVG	MED	MAX
add	20	194	164	361	692	1760	2050	3260	4.41	11.6	9.56	42.2
mul	21	420	514	748	764	2700	2960	4280	3.92	9.10	6.41	42.68157
div	157	635	651	967	902	3210	2830	7320	3.81	6.28	6.39	8.61
sqrt	17	751	730	1070	315	32300	35000	49600	18.0	45.9	43.7	84.5
to string	14	209	182	464	76	852	792	1650	2.58	5.55	4.26	61.18
from string	69	299	299	528	208	982	956	1994	2.67	3.13	2.99	3.93
cmp	15	89	110	274	346	1420	1370	2806	5.66	20.36	12.98	96.3
quantize	74	151	189	203	716	1270	1190	2070	4.27	9.31	8.75	18.1
to int32	9	135	103	227	235	1006	971	1670	3.84	11.0	7.49	100
from int32	5	76	74	160	651	1210	1180	1970	12.1	34.1	15.9	142